Vis Comput
DOI 10.1007/s00371-017-1424-3

ORIGINAL ARTICLE

Patch-based self-adaptive matting for high-resolution image and video

Guangying Cao[1] · **Jianwei Li**[1] · **Xiaowu Chen**[1] · **Zhiqiang He**[2]

Abstract We propose an efficient patch-based self-adaptive matting approach to reduce memory consumption in processing high-resolution image and video. Most existing image matting techniques employ a global optimization over the whole set of image pixels, incurring a prohibitively high memory consumption, especially in high-resolution images. Inspired by "divide-and-conquer," we divide the images into small patches in a self-adaptive way according to the distribution of unknown pixels and handle the small patches one by one. The alpha mattes in patch level are combined according to the weights. Relationships between patches are also considered by locally linear embedding to maintain consistency through the whole image. We also extend the framework to video matting with considering the temporal coherence of alpha mattes. A sampling method is applied to speed up the operation of video sampling. A multi-frame graph model is also proposed to enhance temporal and spatial consistency which can be solved efficiently by Random Walk. Experimental results show that the proposed method significantly reduces memory consumption while maintaining high-fidelity matting results on the benchmark dataset.

Keywords Matting · Patch division · High resolution

✉ Xiaowu Chen
chen@buaa.edu.cn

Jianwei Li
jianwei.li@buaa.edu.cn

[1] State Key Laboratory of Virtual Reality Technology and Systems, School of Computer Science and Engineering, Beihang University, Beijing 100191, China

[2] Lenovo Research, Beijing, China

1 Introduction

Digital matting is to softly extract the foreground object from images which has been applied to a number of important applications such as layer extraction, dehazing, film making. The pixel's color I is a convex combination of foreground F and background B

$$I = \alpha F + (1 - \alpha)B, \tag{1}$$

where α represents the opacity (alpha matte), whose values lie in [0,1] with $\alpha = 1$ indicating a foreground pixel and $\alpha = 0$ denoting a background pixel. However, this problem is highly ill-posed. It is typical to include a trimap or some user scribbles indicating some definite foreground, definite background and unknown region to simplify the problem. At the same time, strong priors will help significantly improve the quality of results.

Existing matting methods have already obtained exciting results. However, with the increasing demand of high-resolution images produced by today's digital cameras, high memory consumption becomes a common problem for matting algorithms. Global optimization-based methods inevitably incur a high cost in memory consumption. For example, one of the best known matting methods [1] requires to allocate about 6 GB memory to process an image with 7.7M pixels, which exceeds the capability of many low-end commodity computers. Matting method for high-resolution images [2] is also explored, but the results are not satisfactory with a low rank on the benchmark [3].

Our key observation is that to achieve efficient and scalable matting on high-resolution images, the operations might be performed on multiple small data inspired by "divide-and-conquer." In this paper, we adopt a self-adaptive division strategy to solve the problem of high memory consump-

Published online: 18 August 2017

tion in image matting. We divide the high-resolution image into small patches, and the shapes of patches are adaptively adjusted according to the distribution of unknown pixels in the trimap. Then, we apply an optimized matting method in each patch in parallel and combine the matting results by a weighted optimization method. With this "divide-and-conquer" strategy, the memory consumption has no relation with the size of image, but only relates to the size of patch, which achieves efficiency and scalability.

One of the problems of this strategy is that some patches may be short of known information. A general situation is that all pixels in a patch are in the unknown area. To address this problem, we merge some adjacent patches close to the unknown patch and redivide the merged region. Another problem is that solving each patch individually will bring edge artifacts between patches. We add relations between patches to tackle this problem. When we build the affinity matrix in one patch, we consider the relationships not only between the pixels in this patch, but also some pixels outside the patch as remedy for nonlocal information of each patch.

We also extend our batch-based matting approach to video matting. However, this cannot be straightforward as the human visual system is highly sensitive to jitter and flickering across frames. This phenomenon is observed when an image matting algorithm is applied directly to every frame of a video sequence. Global optimization-based methods inevitably incur a higher cost in memory and time because the relationships between consecutive frames should be taken into consideration to address temporal incoherence of alpha mattes. To solve these problems, we propose a novel sampling method for video matting to accelerate the operation of sampling. A multi-frame graph model is also proposed to enhance temporal and spatial consistency which can be solved efficiently by Random Walk.

Our main contribution is a patch-based self-adaptive matting framework which processes high-resolution image and video efficiently. A patch division algorithm is presented to divide the image into appropriate patches adaptively according to the unknown pixel distribution. Nonlocal relationships between patches are established to eliminate the edge artifacts. We extend the framework to video matting by introducing a new sampling method and considering the temporal consistency. Experiments on the image benchmark [3] and video benchmark [4] demonstrate that our approach maintains high-fidelity matting results while significantly reducing memory consumption.

The remainder of the paper is organized as follows. We review related work in Sect. 2, followed by description of the proposed image matting approach in Sect. 3. The extension of image matting method to video matting is detailed in Sect. 4. Experimental results are discussed in Sect. 5, and we conclude the paper in Sect. 6.

2 Related work

Existing image matting methods can be roughly categorized into sampling-based methods such as [5–11], propagation-based methods such as [1,12–17], and combination of the two methods such as [9,18]. We only discuss some of the most relevant works here. A more comprehensive survey can be found in [19].

The sampling-based matting methods collect color samples from known foreground and background regions to represent the corresponding color distributions and determine the alpha value of an unknown pixel according to its closeness to these distributions. These methods employ different selection criteria to collect a subset of known F and B samples and different algorithm to search for the best (F, B) pair for each unknown pixel within the representative samples. Once the best pair is found, the final alpha matte is estimated as shown in Eq. (3).

The propagation-based matting methods define an affinity matrix representing the similarity between pixels and propagate the alpha values of known pixels to the unknown ones. These approaches mostly differ from each other in their propagation strategies or affinity definitions. KNN matting [12] defines the affinity matrix by the weights of each unknown pixel with the K-nearest neighbors in the feature space. Some other matting methods such as [1,14] defined the affinity matrix by the weights of the pixels in the local window.

Factually, most methods combine the two methods which will make a good trade-off between the two approaches. Robust matting [9] first collected samples with high confidence and then used the Random Walk [20] to minimize the matting energy constructed by matting Laplacian. The other sampling-based methods [6–8,10,11] also apply local smooth as a postprocessing step to further improve the quality of the estimated alpha matte. The method [18] also combines sampling and propagation which also first collects samples with high confidence and then uses nonlocal and local constraint to propagate the alpha values.

In conclusion, the recent approaches not only the propagation based but also the sampling based need to build a large sparse matrix such as matting Laplacian [1,6–8,13] or other sophisticated matrices [12,14,18,21]. However, solving this kind of large linear system is a very memory-consuming operation, which makes it unpractical to deal with high-resolution images.

Methods for video matting also can be roughly categorized as above. However, to address the problem of spatial and temporal incoherence, most video matting methods use optical flow, such as [22–24], to generate temporally consistent video mattes. However, optical flow can not guarantee to estimate an accurate motion for videos with complex scenes. The optical flow used on high resolution is also a very time-consuming operation. Sampling-based methods

such as [25] extend the sampling area from the current frame to the adjacent frames, which leads to huge sampling set and intolerable time consumption. Propagation-based methods such as [26,27] build multi-frame affinity matrix to guarantee the coherence of mattes, but this will lead to exponentially increase in the time and memory consumption.

Recently, deep learning methods for image matting are developed quickly and achieve good results in the benchmark [28,29]. However, deep learning method is a kind of supervised learning which needs large training data. In this paper, we still focus on traditional matting algorithm without leaning from training images.

This paper extends patch-based image matting presented in the conference version [30] to video matting. We improve the previous work in three aspects. First, we extend our self-adaptive framework to video matting to reduce the memory consumption. Second, a novel sampling method and Random Walk algorithm are adopted to speed up the matting process. Third, lots of experiments on video matting, including qualitative and quantitative comparisons with previous methods, are illustrated to demonstrate the effectiveness of our method.

3 Patch-based self-adaptive matting

We decompose the image into small patches to reduce the memory consumption to handle high-resolution inputs. Theoretically, any existing matting methods could be embedded into our framework in disposing each patch. In this paper, we adopt LNSP method [18] for some reasons. First, the LNSP method keeps in a very high rank in the benchmark for a very long time. Moreover, the nonlocal prior of this method can help our patch-based method by building relations between patches. We optimize the LNSP method in several aspects to fit our patch-based matting framework.

The overview of the patch-based self-adaptive matting is shown in Fig. 1. Given the input image and corresponding trimap (or user scribbles), we first use sampling method, such as comprehensive sampling, to turn some unknown pixels into definite foreground or background pixels and generate the initial alpha matte. Then, the proposed patch division algorithm is applied to divide the unknown region to small patches self-adaptively. After that, we apply matting algorithm to each patch considering both local and nonlocal constraints to make the alpha values of patches consistent. Finally, we combine the alpha matte of all patches together and obtain the matting result of the whole image.

3.1 Comprehensive sampling

To expand the known area in the trimap, we first use sampling algorithm to turn some unknown pixels into definite foreground or background pixels. Here we follow the comprehensive sampling method [6] which produces nice result. We only briefly describe the basic idea. For more details, please refer to [6].

The selection of best pair is done through a brute-force optimization of an objective function which consists of three parts as

$$O_z(F_i, B_i) = K_z(F_i, B_i) \times S_z(F_i, B_i) \times C_z(F_i, B_i), \quad (2)$$

where K indicates chromatic distortion and S and C contain spatial and color statistics of the image, respectively.

Equation (2) estimates the confidence of each candidate pair for pixel i. Finally, we use the pair with the highest confidence to be the best pair that can represent the true foreground and background colors.

Given a selected foreground and background sample pair (F_i, B_i) for pixel i, the alpha value of pixel i can be estimated as

$$\alpha_i = \frac{(C_i - B_i)(F_i - B_i)}{\|F_i - B_i\|^2}. \quad (3)$$

After sampling, the initial alpha values and the corresponding confidence map are obtained. In the unknown area, alpha values with high confidence can be seen as definite known pixels. The confidence threshold is set to be 0.2 in all our experiments. The problem of the sampling process is that it is very time-consuming to conduct on high-resolution images directly. Thus, we scale the image into a smaller one to do the sampling and rescale the sampling result to the original resolution. In our experiments, we scale the image

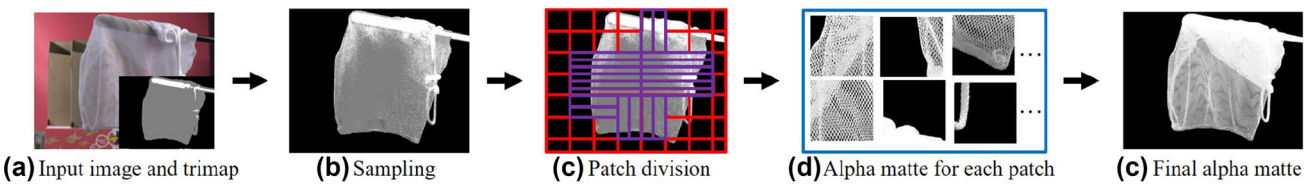

Fig. 1 Overview of patch-based self-adaptive matting. Given the input image and corresponding trimap (**a**), we first use sampling method (**b**) to turn some unknown pixels into definite foreground or background pixels. Then, patch division algorithm (**c**) is applied to divide the unknown region to small patches. After that, we apply matting algorithm to each patch considering both local and nonlocal constraints (**d**). Finally, we combine the alpha matte of all patches together and obtain the matting result of the whole image (**e**)

into no more than 800 pixels in both width and height. For example, the image *doll* with 19.31% unknown pixels takes about 90 min in sampling process on original resolution (3151 × 2224) and only takes 6 min on the smaller one (800 × 565). We find that sampling on the smaller image can also produce good result while saving time cost.

3.2 Self-adaptive patch division

After sampling process, the definite alpha values are going to be propagated to the unknown area. To reduce memory consumption, we divide the high-resolution image into overlapped patches. In our experiment, the patch size is 100 × 100 by default, and the adjacent patches are 20 pixels overlapped. The patches without unknown pixels are regarded as known patches and do not need to be computed again. However, some divided patches encounter large unknown area and may lack enough information of known foreground or background pixels. This kind of patch will result in very bad alpha matte. For example, the patches labeled *a* and *b* in Fig. 2d only have unknown pixels. We call this kind of patch as low-confidence patch. If the percentages of two kinds of known pixels (foreground and background) are both larger than 5% in one patch, this patch will be seen as high-confidence patch. Otherwise, the patch will be regarded as low-confidence patch.

For a low-confidence patch i, we would like to merge it with its adjacent patches to increase the confidence. We have the observation that if the quantities of three kinds of pixels (foreground, background, unknown pixels) have large difference between this patch and its neighbor, the two patches are suitable to be merged to get a higher confidence. Under this observation, we define a variance using pixel quantity in different region of adjacent patches to determine which patches are merged and redivided. There are two merging direction: horizontal and vertical. The variance of patch i is defined as

$$\begin{aligned} \text{variance}_i^h &= \sum_{j\in\{f,b,u\}} \parallel N_{ij}^{h_1} - N_{ij}^{h_2} \parallel, \\ \text{variance}_i^v &= \sum_{j\in\{f,b,u\}} \parallel N_{ij}^{v_1} - N_{ij}^{v_2} \parallel, \end{aligned} \tag{4}$$

where N represents the number of different kinds of pixels, f, b, u represent the foreground, background and unknown region, respectively. The notations h, v represent horizontal and vertical direction, respectively, and h_1 and h_2 represent the horizontal adjacent patches, v_1 and v_2 represent the vertical adjacent patches.

As shown in Fig. 2, if the variance of horizontal adjacent patches is greater than vertical adjacent patches, the merging direction is horizontal, and vice versa. Once the direction is determined, the adjacent patches will be merged along this

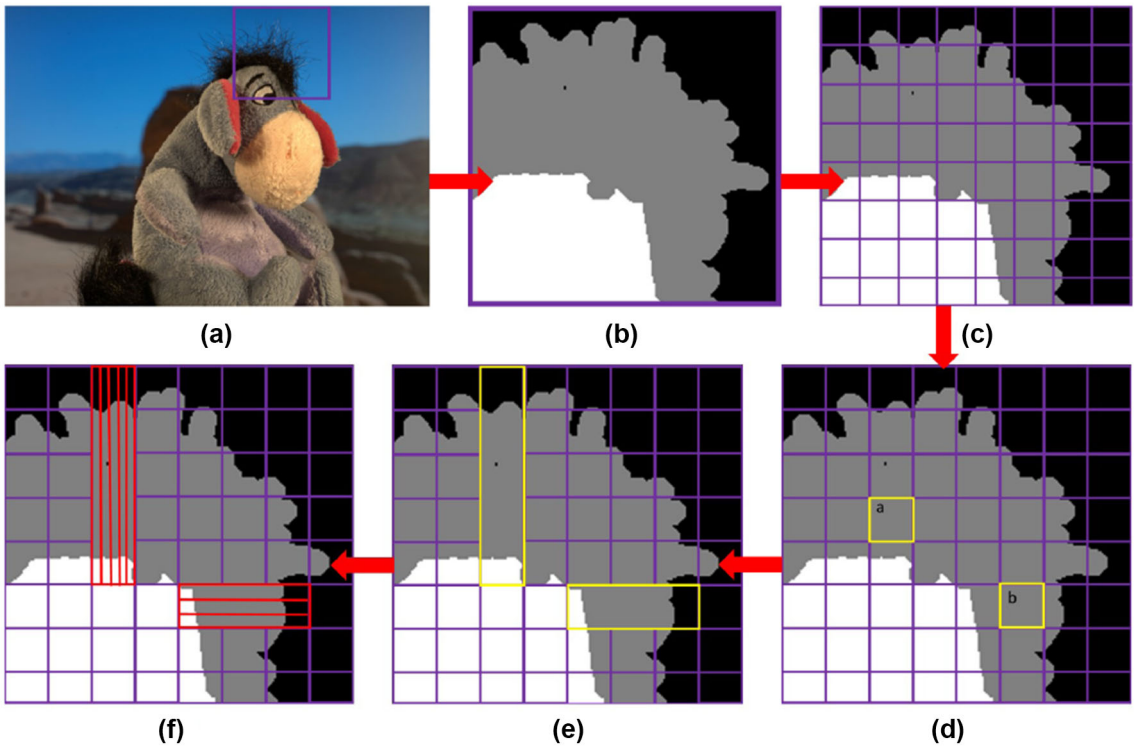

Fig. 2 Process of patch division. **a** Original image; **b** cropped trimap; **c** dividing trimap into patches; **d** bad patches estimation; **e** patches merging; **f** redividing merged regions

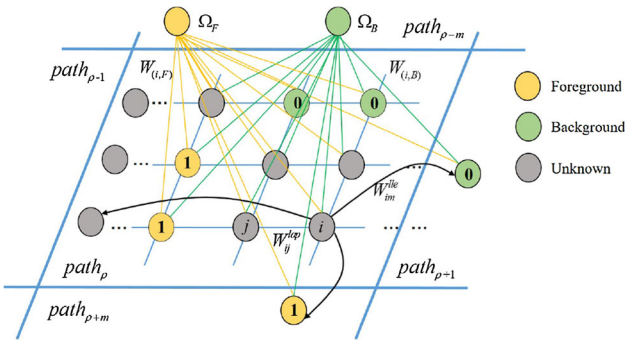

Fig. 3 Graph model of one patch. Two virtual nodes are connected with all the pixels in this patch and the selected neighbors outside this patch. Each pixel is also connected with its spatial neighbors in local term and feature space neighbors in nonlocal term

direction. The merge process will continue until the merged patches have enough foreground and background known pixels or encounter known patch. To guarantee small computing scale, the region is redivided into new patches containing about 10,000 pixels. As shown in Fig. 2f, if the merging direction is horizontal, the merged region will be redivided into the same amount of vertical patches, and vice versa. The redivided patches have enough known pixels and a high confidence. Figure 2 shows the process of patch division. When all the patches are in a high confidence, it is time to compute the alpha matte patch by patch.

3.3 Graph model

After patch division, we apply the optimized version of the LNSP method [18] to each patch. An optimized graph model is built for each patch as shown in Fig. 3. Our graph model includes nonlocal term, local term and data term. We take patch ρ to describe the graph model in this section.

3.3.1 Nonlocal term

Each patch has lost nonlocal information since there is no relation between divided patches. Thus, we build relationships between patches by finding K nearest neighbors for each pixel from the whole image in the global RGBXY feature space, which is considered as a remedy for nonlocal information. However, the number of neighbors of all pixels in one patch is much larger than that in the patch itself, which results in large extra memory consumption. Besides, the weak constraint of unknown pixels outside the patch will lead to inaccurate alpha matte. Thus, we only select the neighbors inside this patch and the neighbors with known alpha values outside this patch. Here the known pixels include not only the known pixels labeled by the trimap or user scribbles, but also the pixels with high confidence according the confidence map obtained by comprehensive sampling at the first step. After selection, the quantity of all neighbors outside this patch is controlled in about 100–200. The memory consumption will increase only about 10–20% due to the nonlocal information, but the quality of result will obtain a great improvement.

We connect the pixel X_i to the selected neighbors $X_{i1}, X_{i2}, \ldots, X_{iD}$ with weights W_{id}^{LLE}, where D represents the number of selected neighbors. As stated in [31], the W_{id}^{LLE} can be computed by minimizing

$$W^{\mathrm{LLE}} = \arg\min_{W_{id}^{\mathrm{LLE}}} \sum_{i=1}^{N} \parallel X_i - \sum_{d=1}^{D} W_{id}^{\mathrm{LLE}} X_{id} \parallel^2 . \tag{5}$$

The W^{LLE} will be an $N \times H$ matrix, where N represents the number of pixels in the patch and $(H - N)$ is the number of selected neighbors outside this patch. The last $(H - N)$ columns in W^{LLE} are the weights between the selected pixels outside the patch and the pixels in the patch.

3.3.2 Local term

The local matting Laplacian enhances the local smoothness of the alpha matte. W^{Lap} is defined as: $W^{\mathrm{Lap}} = [W_{in}^{\mathrm{Lap}}, \mathbf{0}]$, where W_{in}^{Lap} is the matting Laplacian of $N \times N$ representing the weights between pixels and neighbors in a 3×3 window in this patch. For more details about matting Laplacian, please refer to [1]. And $\mathbf{0}$ is a zero matrix of $N \times (H - N)$.

3.3.3 Data term

For pixel i in one patch, its data weights, $W_{(i,F)}$ and $W_{(i,B)}$ which represent the probability of a pixel belonging to foreground and background, respectively, are defined between each pixel i and two virtual nodes F, B to enforce the data constraint. For pixel i, the data weights are defined as

$$W_{(i,F)} = \gamma \alpha_i, \quad W_{(i,B)} = \gamma (1 - \alpha_i). \tag{6}$$

The initial alpha values of this patch in Eq. (6) are the results by comprehensive sampling before using local smooth proposed in [6]. Here, γ is set to 0.5. Besides the pixels inside this patch, the pixels connected to the visual nodes also include the selected neighbors in W^{LLE} outside this patch.

3.3.4 Closed-form solution for each patch

For the patch ρ, the energy function for solving alpha values is defined as

$$E_\rho = \lambda \sum_{i \in \Omega} (\alpha_i - g_i)^2 + \sum_{i=1}^{H} \left(\sum_{i \in N_i} W_{ij} (\alpha_i - \alpha_j) \right)^2 , \tag{7}$$

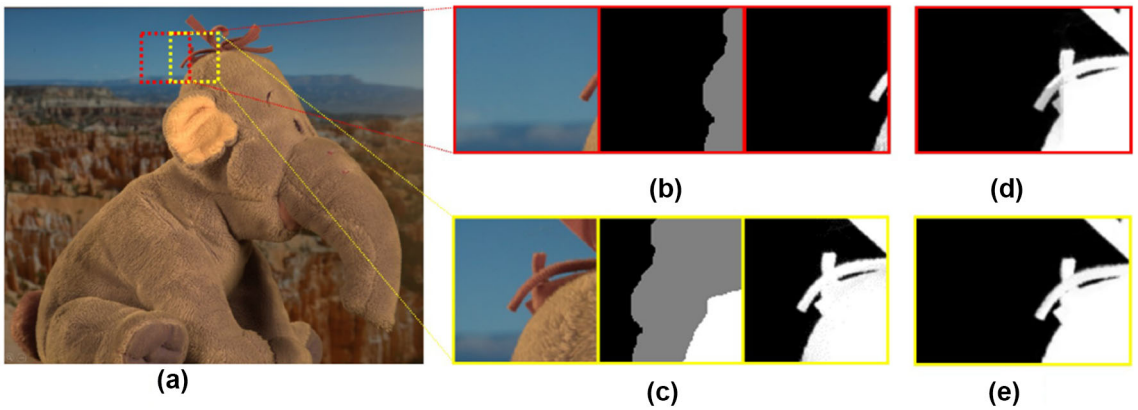

Fig. 4 Comparison of overlap fusion. **a** Is the input image; **b** and **c** are the overlapped patches including the cropped image patch, trimap and alpha matte; **d** is the direct average of overlapped patches; **e** is the weighted average of overlapped patches

where the set Ω includes the visual nodes, the known pixels in patch ρ after sampling, and the selected pixels as neighbors outside patch ρ. H is the total number of the nodes in the graph model associated with patch ρ. W_{ij} is the sum of three kinds of weights, containing nonlocal weights W_{ij}^{LLE}, local weights W_{ij}^{Lap} and data weights $W_{(i,F)}$ and $W_{(i,B)}$. N_i is the set of neighbors of pixel i, including neighboring pixels in 3×3 window, the selected nonlocal neighbors in feature space and two virtual nodes. Equation (7) can be further written in a matrix form as

$$E_\rho = (\alpha - G)^T \Lambda(\alpha - G) + \alpha^T L^T L\alpha, \tag{8}$$

in which

$$L_{ij} = \begin{cases} W_{ii}, & \text{if } i = j \\ -W_{ij}, & \text{if } i \text{ and } j \text{ are neighbors.} \\ 0, & \text{otherwise} \end{cases}$$

Here, the weight $W_{ii} = \sum_{i\in N_i} W_{ij}$. Λ is a diagonal matrix of $H \times H$ and Λ_{ii} is 1000 if $i \in \Omega$ and 0 otherwise. G is a vector of dimension H, and G_i is set to 1 if i belongs to foreground and 0 otherwise.

Equation (8) is a quadratic function about α, which can be minimized by solving the linear equation in closed-form solution

$$(\Lambda + L^T L)\alpha = \Lambda G. \tag{9}$$

The first N values in the α vector are the alpha values of the pixels in the patch ρ. It is obvious that the computation of all patches is independent of each other. Thus, the above operations can be performed in parallel. This way could greatly reduce the time consumption.

3.4 Global alpha matte

Since the alpha matte of every patch has been obtained, the goal in this section is to get the global consistent alpha matte. The main problem is how to compute the alpha values in the overlapped region of those adjacent patches and eliminate the edge artifacts. To get more accurate alpha values, we give the patches different weights according to the number of different kinds of pixels. The weight is defined as

$$W_\rho = e^{-\frac{(P_f-P_b)^2+(P_f-P_u)^2+(P_b-P_u)^2}{\sigma_1}} \times e^{-\frac{P_u}{\sigma_2}}. \tag{10}$$

Here, $P_d, d = \{f, b, u\}$ represents the percentage of different kinds of pixels. The σ_1 and σ_2 are constants trading off the effect of variance and the quantity of unknown pixels. The overlapped patches are blended according to their weights to generate the final result. Equation (10) is based on the observations: The alpha matte will be more accurate if this patch has enough foreground and background pixels and not too many unknown pixels. We use the variance to measure the number of pixels in different region. If the patch only has little foreground, the alpha values of unknown pixels might be much smaller. If the patch only has little background, it might result in oversmooth. The direct average alpha values in the overlapped region may lead to edge artifacts, as shown in Fig. 4d, while weighted average alpha values will be more accurate, as shown in Fig. 4e.

4 Extension to video

In this section, the proposed patch-based image matting framework is extended to video to extract spatially and temporally consistent alpha mattes. There are two main improvements in patch-based video matting. First, we optimize comprehensive sampling algorithm in Sect. 3.1 to make

it faster and better in performance. Second, we extend the graph model in Sect. 3.3 across consecutive frames to consider temporal consistency.

4.1 Sampling algorithm for video

We introduce a new sampling method for video matting to reduce the time consumption. When processing high-resolution images or videos, the number of samples in the comprehensive sampling set can be very huge which leads to large time consumption. Inspired by [6], the sampling range can be varied according to the distance of a given pixel to the known foreground and background. To do the sampling fast and efficiently, we reduce the sampling range of the given pixel. Specifically, we select the known foreground and background samples around the boundaries of known region near the given unknown pixel to constitute the local sampling set.

For each pixel i in unknown region, we construct a corresponding local sampling set including known foreground and background pixels. For each possible foreground and background sample pair, we calculate the alpha value α of pixel i by Eq. (3). Then, we get the reconstruction error by $\epsilon = \|C_i - (\alpha F_i + (1-\alpha)B_i)\|$ to measure the confidence of the alpha value. We select the alpha value which has the lowest reconstruction error as the best one.

However, shrinking the sampling set to the local one brings a problem that the selected sample pair may not be the best one in a global view. Therefore, we learn representative and discriminative dictionaries from the entire known region of the video to involve the global information. Furthermore, the global dictionary can represent every frame of the video and provide consistency between frames. We learn two dictionaries D_F, D_B from the foreground and background known region, respectively. We conduct down-sampling on the key frames to remove some redundant information of high-resolution images. This operation could largely reduce the training time of the initial dictionary learning. Following the method in [32], we firstly initialize two sub-dictionaries by sparse coding algorithm [33] according to foreground and background known pixels. The two sub-dictionaries D_F and D_B represent foreground feature and background feature, respectively. Then, we use the KL-divergence to eliminate the redundant samples in the initial dictionary $D = \{D_F, D_B\}$. By maximizing the KL-divergence between each pair of atoms in D, the cross-correlation between atoms is minimized to make the dictionary representative and discriminative. The dictionary D can be regarded as a global sampling set, which is comprehensive enough to represent the entire known region and small enough to reduce the time consumption of coding. We can use the sparse coding on the global dictionary to estimate the alpha values of pixels in the unknown region by the method mentioned in [10]. The sum of the normalized sparse codes of corresponding foreground samples in the dictionary directly provides the α value. We also calculate the reconstruction error to measure the confidence of the alpha value, like the local sampling step above.

Compared to previous dictionary-based video matting [26], we use KL-divergence to make the dictionary become smaller with better representative ability. We also use local sampling set to supplement the global dictionary set to achieve a better initial alpha matte. For each pixel in the unknown region, we get two alpha values with their confidences in local and global reconstruction. We choose the alpha value with higher confidence (smaller reconstruction error) as the final α at each pixel. As shown in Fig. 5, the proposed sampling method can obtain the good initial alpha values with the help of local and global sampling. For an example shown in Fig. 5, the mean of absolute differences (MAD) is about 0.0646 only by local sampling and 0.0758 only by global sampling. However, the MAD reduces to 0.0424 if we combined local sampling with global sampling. The result by proposed method is better than the result produced by [6] (the MAD of the example is 0.0590), and the time consumption is largely reduced in the mean while.

4.2 Multi-frame graph model

In video matting, the graph model is extended to multi-frame involving temporal domain. Thus, the multi-frame graph model can make the alpha mattes more consistent between frames. We process two adjacent frames each time to generate the multi-frame graph model. The two adjacent frames are divided into the same size patches, and the multi-frame graph model is generated and solved in the overlapped patches. The patch division process is similar to that in image matting.

There are two differences between multi-frame graph model and image graph model. First, in nonlocal constraint, the feature space becomes RGBXYT, where T represents time domain. Thus, the nearest neighbors can also come from the adjacent frame. Second, in local constraint, the 3D matting Laplacian is applied in a $3 \times 3 \times 2$ window. Both the two improvements consider temporal information to make the matting results consistent in time domain. In addition, the equations and construction steps are the same with those in Sect. 3.

However, due to the additional time domain, there exists a problem that the multi-frame graph model will bring larger memory consumption when solving Eq. (8). To solve this problem, we introduce the Random Walk [20] to reduce the memory and time consumption. In [20], the graph labeling problem is regarded as a Random Walk since the alpha values are continuous. We decompose L into blocks corresponding

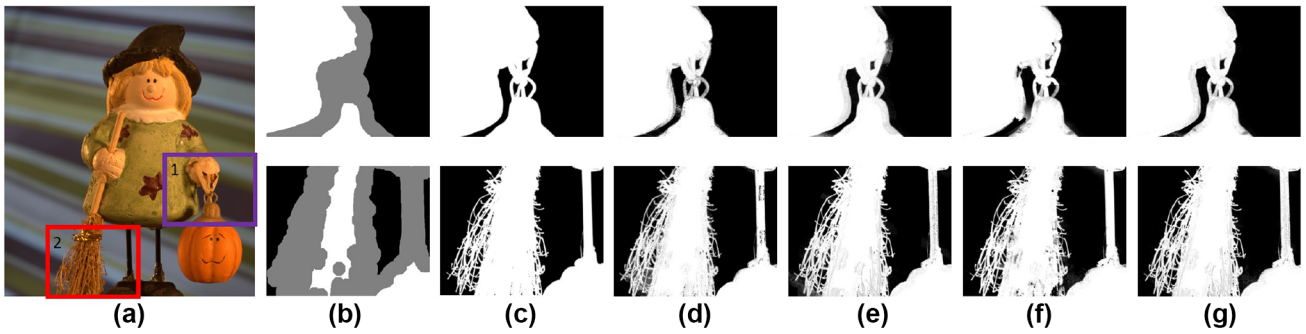

Fig. 5 Qualitative comparison of proposed sampling. **a** Input image; **b** trimap; **c** groundtruth; **d** result by method [6]; **e** result only using global sampling; **f** result only using local sampling; **g** result using local and global sampling

to unknown pixels and known pixels including user labeled pixels and virtual nodes, as

$$L = \begin{bmatrix} L_k & R \\ R^T & L_u \end{bmatrix}. \tag{11}$$

where L_k and L_u are weights corresponding to known and unknown pixels, respectively, and R represents others. It has been shown in [20] that the probabilities of unknown pixels belonging to a certain label (for example, foreground) are the solution to

$$L_u A_u = -R^T A_k, \tag{12}$$

where A_u is the vector of unknown values we need to solve for and A_k is the vector encoding the boundary conditions (i.e., 1s and 0s for the known foreground and background, respectively). L_u is guaranteed to be nonsingular for a connected graph; thus, the solution A_u is guaranteed to exist and be unique with values guaranteed to lie between 0 and 1. Equation (12) can be calculated directly with matrix manipulation.

5 Experiments

We test the proposed method on various images and videos and compare it to state of the art to show the effectiveness achieved by our matting method and evaluate the performance of the proposed method on benchmarks including image benchmark dataset [3] and video benchmark dataset [4], which are proposed in the paper [34,35], respectively.

The image benchmark dataset contains 35 natural images. Among those images, 27 of them constitute the training set where the groundtruth alpha mattes are available. The remaining eight images without groundtruth alpha matte are used for evaluation. The video benchmark dataset provides ten videos with three kinds of trimaps to evaluate the performance. Both of the benchmarks provide high-resolution images or videos to evaluate.

Table 1 Memory and time comparison of different methods on different resolution images

Performance comparison for image matting				
	Image	*Plant*	*Elephant*	*Plasticbag*
	Resolution	667 KB	8.83 MB	9.14 MB
	Unknowns	23.17%	9.74%	24.12%
CF [1]	Memory	25.15 MB	196.78 MB	405.92 MB
	Time	0.8 min	2 min	4 min
KNN [12]	Memory	50.26 MB	954.31 MB	958.58 MB
	Time	0.7 min	2 min	4 min
LB [14]	Memory	25.15 MB	196.78 MB	405.92 MB
	Time	0.7 min	2 min	4 min
CS [6]	Memory	25.15 MB	196.78 MB	405.92 MB
	Time	8 min	54 min	115 min
Our	Memory	2.35 MB	0.42 MB	2.55 MB
	Time	10 min	18 min	62 min

5.1 Memory and time comparison

In image matting, we choose four representative methods to compare with our approach. The source codes of these methods are available on the internet. Table 1 lists the comparison of memory and time consumption with these methods. Only three examples (*plant, elephant, plasticbag*) are listed for saving space. Here we only illustrate the memory cost by large sparse matrices, which are the highest memory cost during solving process. Since we divide the images into patches, we only build a sparse matrix based on patch instead of the whole image. Because patches differ from each other, we only consider the average number of elements of the patches.

As shown in Table 1, the methods using the matting Laplacian such as [1,6] or other sophisticated matrices such as [12,14] consume large memory, while the proposed method consumes little memory which is only about 0.1–1% of them. This is because the memory consumed by proposed method relies on the size of the patch and the percentage of unknown pixels and has little relation with the image resolution.

Table 2 Ranks of different matting methods with respect to the three measurements on image benchmark dataset [3]

SAD		MSE		Gradient	
Method	Rank	Method	Rank	Method	Rank
1. LNSP	10.0	1. LNSP	8.6	1. GS	9.7
2. GS	10.3	2. Ours	9.4	2. Ours	9.9
3. Ours	10.3	3. KL-Sparse	10.8	3. KL-Sparse	10.0
4. KL-Sparse	10.9	4. CCM	11.1	4. LNSP	10.3
5. TSPS-RV	11.9	5. GS	11.5	5. CS	11.1
6. IT	12.6	6. TSPS-RV	12.0	6. CCM	13.2
7. CS	12.8	7. CS	12.2	7. SVR	13.2
8. SVR	13.2	8. SVR	12.5	8. SC	13.3
9. CWCT	13.3	9. CWCT	13.3	9. Segment	13.8
10. SC	13.7	10. CSC	14.5	10. Global	13.9

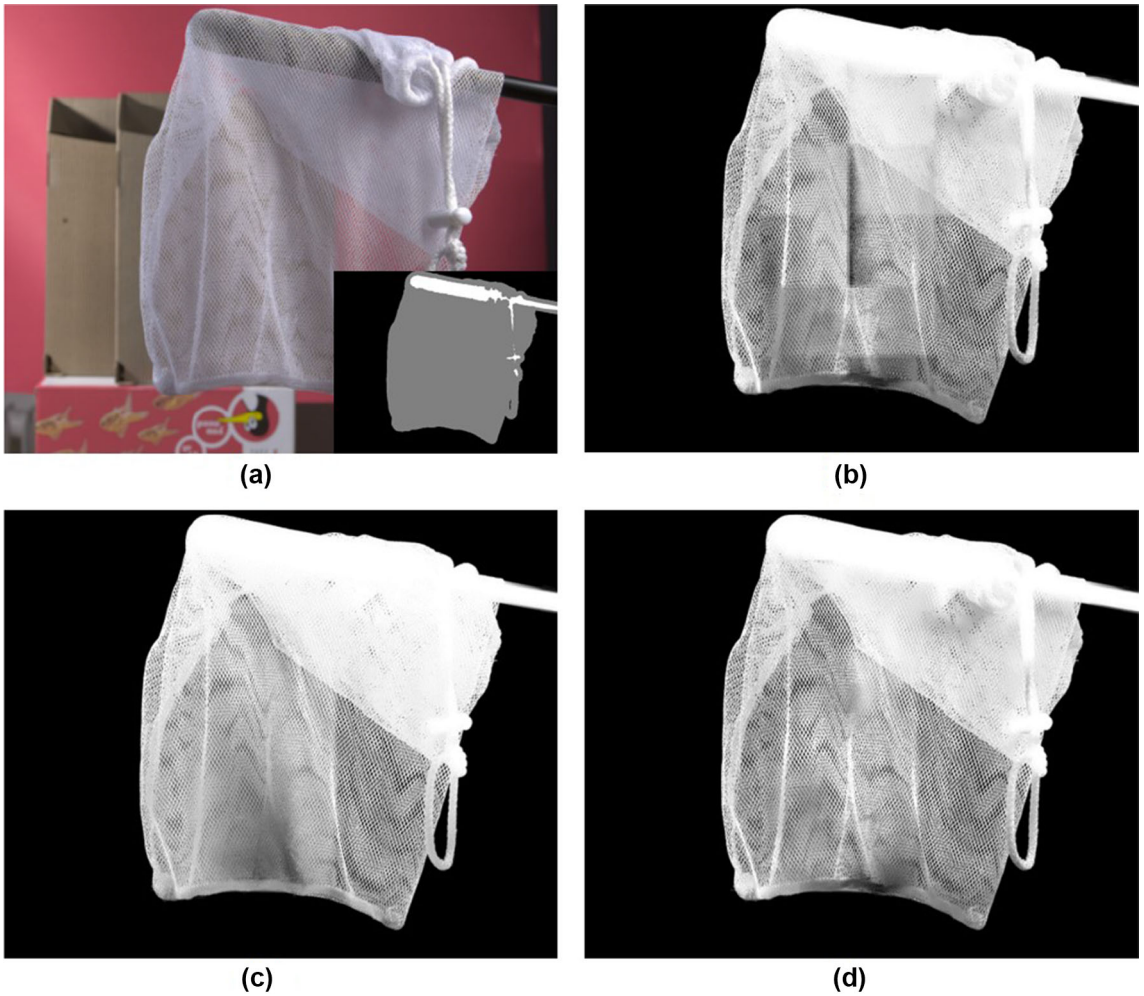

Fig. 6 Comparison with LNSP method [18]. **a** The original image and trimap; **b** the result directly using LNSP in each patch; **c** the LNSP result for the whole image; **d** our patch-based result

The sampling stage in image matting is very time-consuming on high-resolution images. Thus, in video matting, we apply our new sampling method to get the initial alpha values, which reduce time consumption significantly. It takes about 15 min to process each frame of the example *vitaliy* which contains about 2.1 million pixels and 8.83% unknown pixels by the comprehensive sampling method [6]. But it only takes less than 1 min to get the sampling result

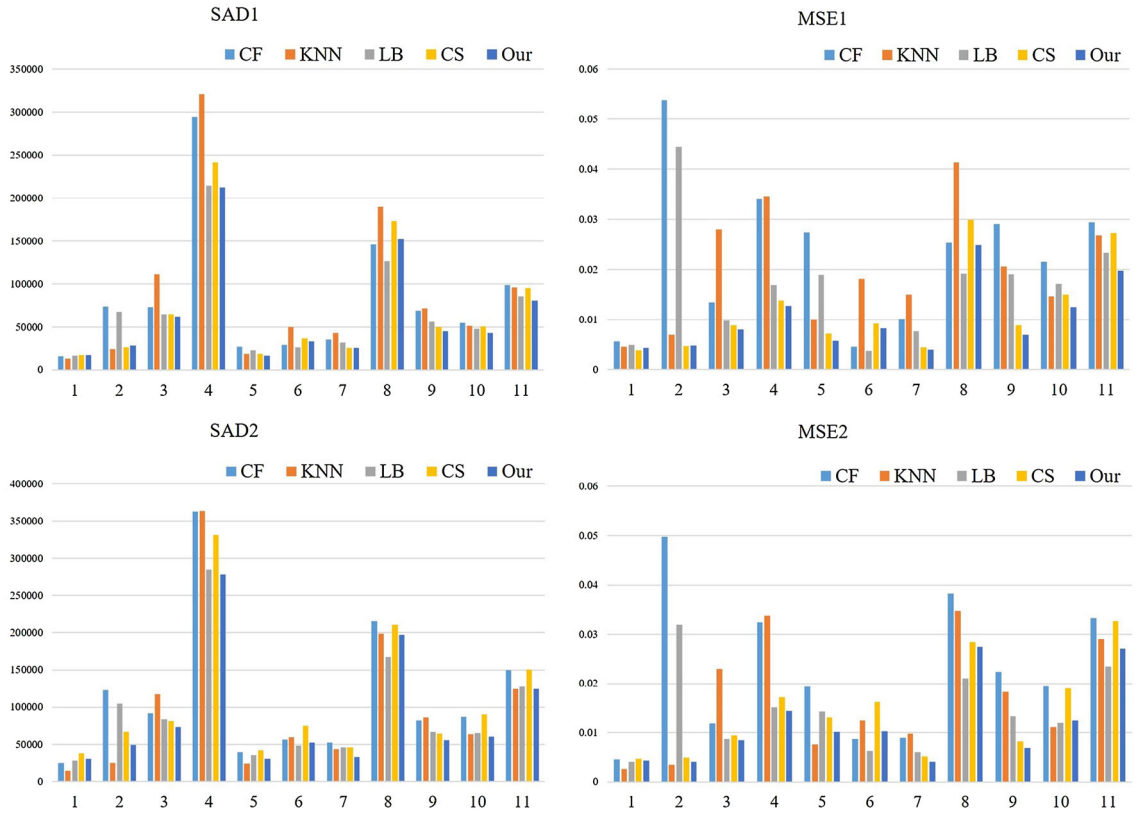

Fig. 7 Evaluation compared with CF [1], KNN [12], LB [14] and CS [6] on the high-resolution image dataset with different trimaps for MSE and SAD measurement

by the proposed sampling method. The total time by the proposed method is closed to 7 min. The example *flower* containing about 0.25 million pixels and 24.46% unknown pixels consumes about 15 min by [6], while it only takes about 1 min to finish the sampling operation and 6 min to get the final result by the proposed method. The time consumption decreases greatly, and the results are comparable with the results produced by [6]. Since we take the information of adjacent frames into consideration to enhance spatial and temporal coherence when we process video, the memory consumption on video matting may be about three times larger than that on single image. However, the proposed method still has great advantage in terms of reducing memory.

5.2 Evaluation on image matting

The most advantage of the patch-based matting method compared to LNSP matting method is that the proposed method is able to process high-resolution images with low memory consumption, while LNSP matting method cannot do that. When processing high-resolution image such as 1920×1080, the LNSP matting method is often out of memory in consumer computer with 8 GB RAM. At the same time, the performance of the matting results of the proposed method and LNSP method is almost the same, as shown in Fig. 6c, d. The ranks of the two methods in the benchmark as shown in Table 2 are also very close.

The nonlocal constraint and weighted combination algorithms are the keys to make the alpha result consistent at the whole image. As shown in Fig. 6, we directly use LNSP matting method in each patch without nonlocal constraint or weighted combination. We can see that the result (b) exists obvious edge artifacts because the alpha mattes in different patches are not consistent. The result (d) by the proposed method removes edge artifacts very well.

Since the benchmark cannot evaluate on the high-resolution dataset, we compare the proposed method with other methods on the low-resolution dataset evaluated by [34]. Here we only show the top ten methods for each measurement. The proposed method ranks at the top three as shown in Table 2. The test result has been published on the benchmark [3].

We also use 27 high-resolution images with groundtruth to compare the proposed method with CF [1], KNN [12], LB [14] and CS [6] on different trimaps which has different width of unknown region. Two main different metrics, namely SAD (sum of absolute differences) and MSE (mean squared error), are taken into consideration. As shown in Fig. 7, SAD1 and MSE1 are the testing results on the first

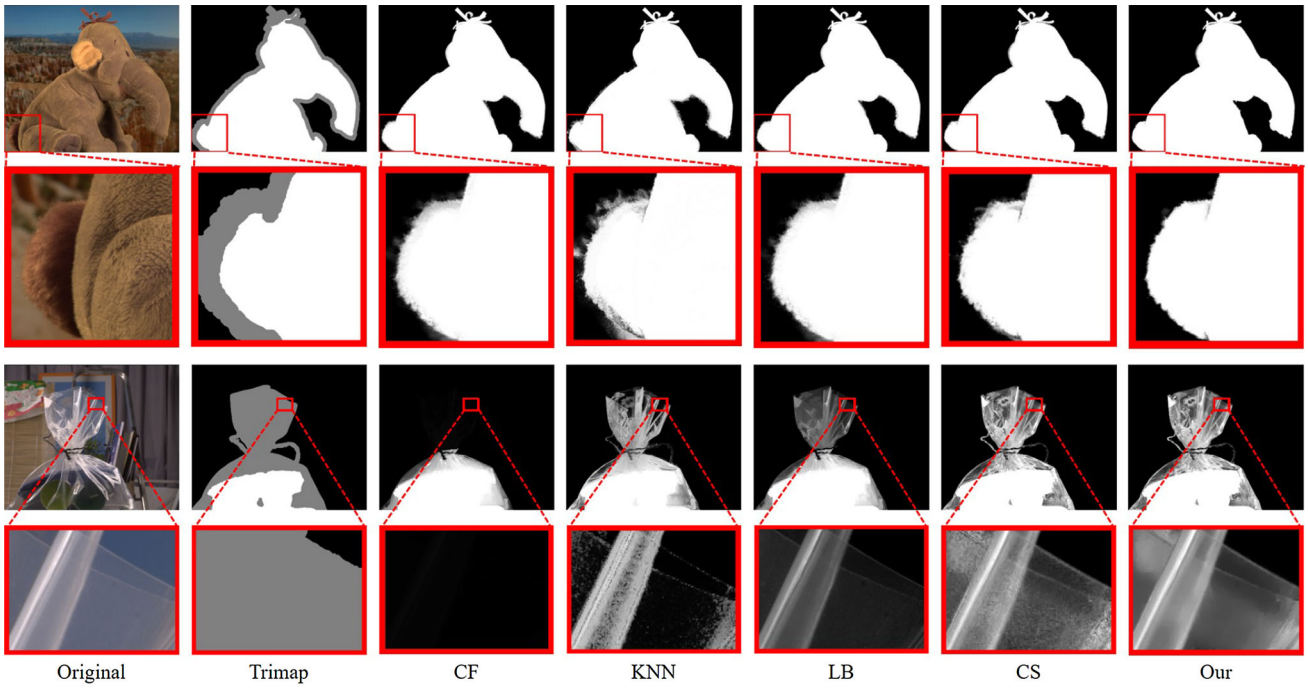

Fig. 8 Visual comparison of the proposed method with CF [1], KNN [12], LB [14] and CS [6] on the high-resolution image testing dataset

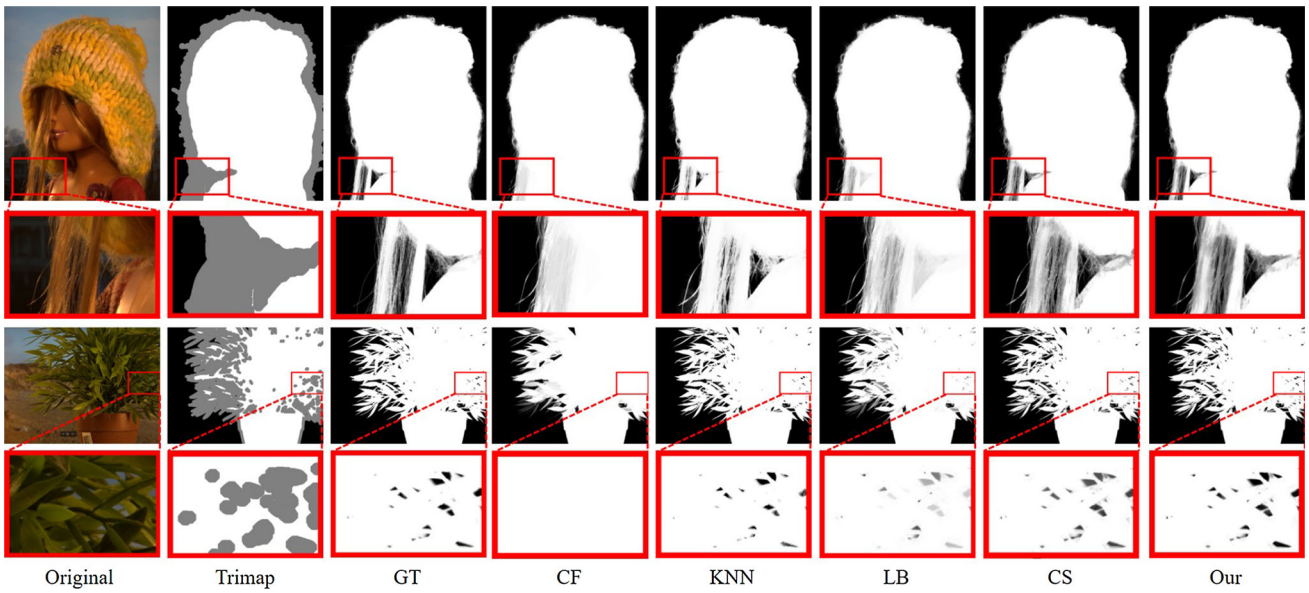

Fig. 9 Visual comparison of the proposed method with CF [1], KNN [12], LB [14] and CS [6] on the high-resolution image dataset with groundtruth

kind of trimap, and SAD2 and MSE2 are the testing results on the second kind of trimap. Being limited by the space, we only show the evaluation of first ten high-resolution images. The last column in Fig. 7 is the average SAD and MSE on different trimaps. From the figure, we can see that the proposed approach outperforms other methods on high-resolution images.

We show two examples without groundtruth to compare the visual quality of alpha matte with CF [1], KNN [12], LB [14] and CS [6]. The foreground and background in the example *elephant* have similar color distribution. Figure 8 shows that the proposed method well distinguishes foreground and background from the similar color distribution. But the edges of foreground and background by other four methods are

Table 3 Ranks of different matting methods with respect to the three measurements on video benchmark dataset [4]

SSDA		dtSSD		MESSDdt	
Method	Rank	Method	Rank	Method	Rank
1. Ours	2.8	1. Ours	3.8	1. Ours	3.5
2. LB	3.8	2. LB	4.0	2. CF	3.5
3. CF	5.6	3. CF	4.1	3. LB	3.6
4. CS	5.7	4. RE	5.2	4. RE	5.9
5. Robust	6.1	5. Robust	6.0	5. CS	6.6
6. Shared	6.7	6. CS	6.6	6. Robust	7.2
7. RE	7.7	7. Shared	7.3	7. Shared	7.6
8. Bayesian	9.8	8. KNN	10.7	8. KNN	10.3
9. KNN	10.9	9. Bayesian	11.0	9. Bayesian	11.0
10. Nonlocal	11.7	10. Spectral	11.2	10. Spectral	11.6

Table 4 Comparisons of error rates with CF [1], KNN [12], LB [14] and CS [6] on four videos

	CF [1]	KNN [12]	LB [14]	CS [6]	Ours
juneau	0.2381	0.4726	0.3120	0.2617	0.2316
slava	0.0723	0.0856	0.0754	0.0763	0.0614
snow	0.0302	0.0425	0.0304	0.0292	0.0250
vitaliy	0.1018	0.1461	0.1010	0.0917	0.0887

blurry. The second example *plasticbag* has large semitransparent region. Figure 8 shows that the alpha mattes produced by other four methods are not consistent but ours is consistent and smooth.

We also illustrate two examples with groundtruth in Fig. 9. The first example has tiny hair, and there are many holes among the hair. The methods of CF [1] and LB [14] fail to distinguish the tiny hair, CS [6] takes some foreground as background, and KNN [12] takes some background as foreground when they process the tiny hair. However, the proposed method gets accurate alpha matte of the tiny hair. The second example has many small holes. The methods CF [1], KNN [12], LB [14] and CS [6] produce oversmooth results when they process the small holes. The proposed method successfully extracts the small holes and produces similar result with the groundtruth as shown in Fig. 9.

5.3 Evaluation on video matting

We evaluate our video matting method on the benchmark dataset with other methods. Here we only show the top ten methods for each measurement. As shown in Table 3, our method ranks at the first place on all three metrics in the narrow trimap. The evaluated result has been published on the benchmark [4].

We also evaluate the temporal coherence by measuring differences of alpha values between successive frames according to [36]. The measure of the difference $\mathrm{dif}(i)$ for the pixel i in frame t is defined as

$$\mathrm{dif}(i) = \frac{\alpha_i(t+1) - \alpha_i(t)}{I_i(t+1) - I_i(t)}, \tag{13}$$

where $\alpha_i(t)$ represents the alpha value of pixel i in frame t, and its RGB color feature is denoted by $I_i(t)$.

Table 4 shows the error rate on four videos of our matting method compared with CF [1], KNN [12], LB [14], CS [6]. These four videos are *juneau*, *slava*, *snow* and *vitaliy*. In the table, we only show the average alpha difference $\sum_{t=1}^{n}\sum_{i=1}^{m}\mathrm{dif}(i)/(m \times n)$ of the whole video. Obviously, the proposed method generates more coherent results on each video than the other four methods. Figure 10 shows the alpha difference of every frame of the proposed method compared

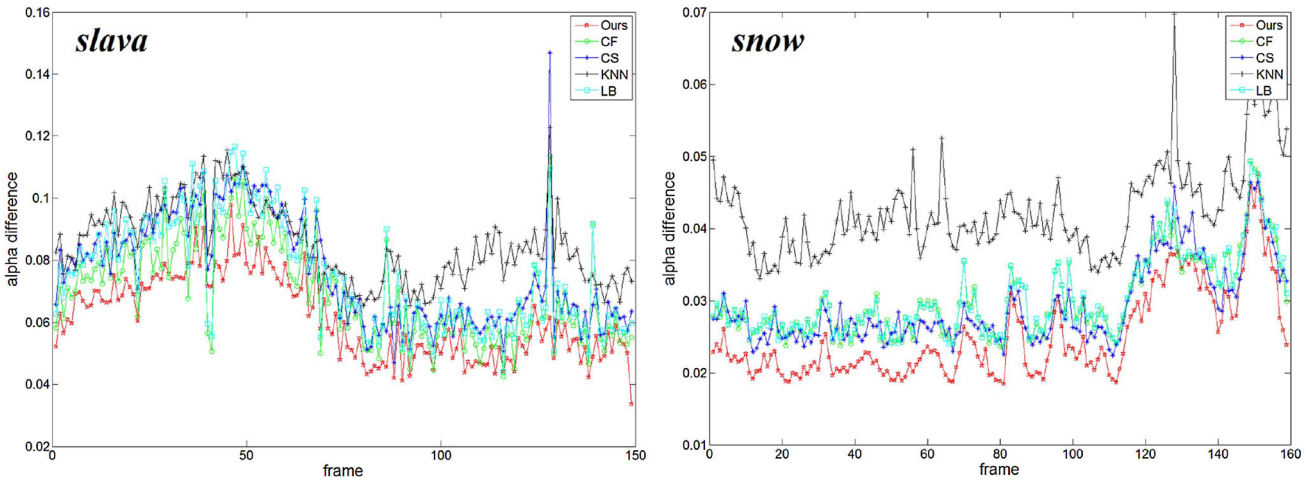

Fig. 10 Comparisons of temporal coherence with CF [1], KNN [12], LB [14] and CS [6] on the video *slava* and *snow*

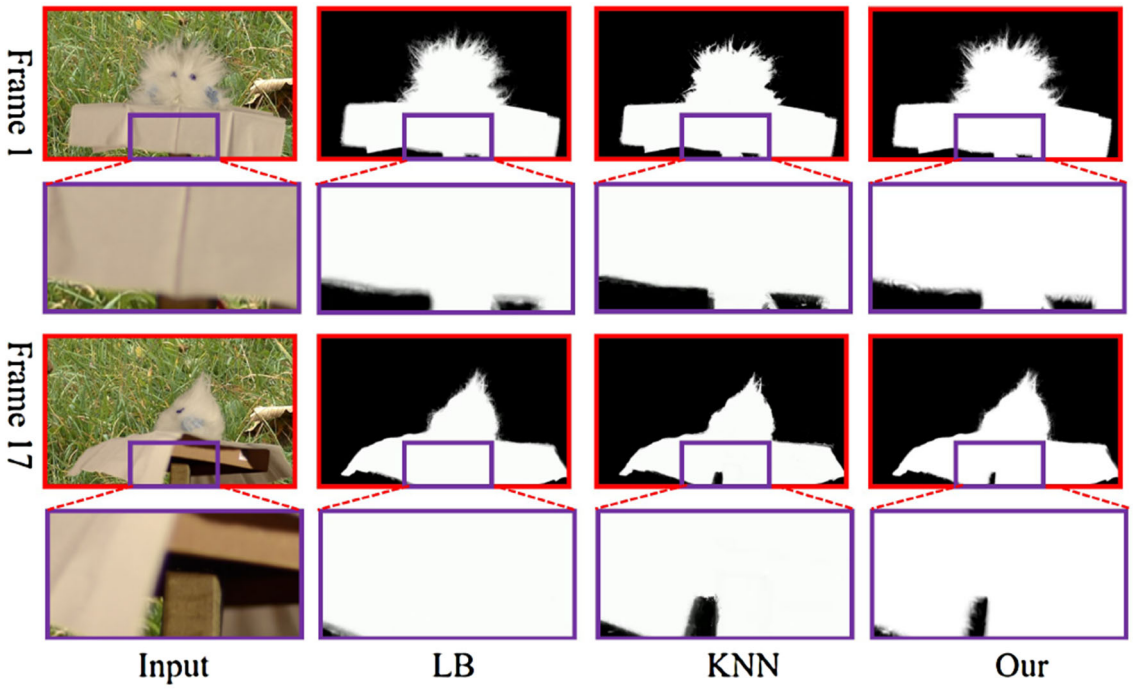

Fig. 11 Comparisons with LB matting [14] and KNN matting [12] on the video *slava*

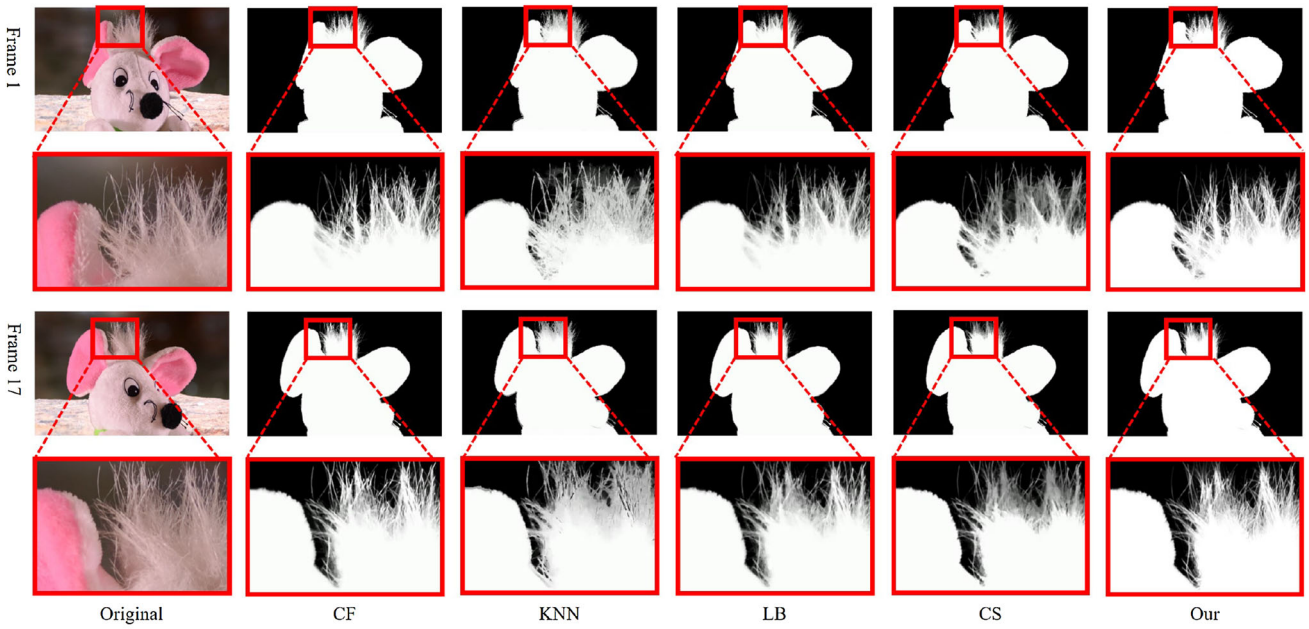

Fig. 12 Comparisons with matting methods CF [1], KNN [12], LB [14] and CS [6] on the video *vitaliy*

to other four methods. From this figure, we can see that the proposed method achieves good temporal consistency than related methods.

Here we demonstrate some examples to show the visual comparison with other methods. Figure 11 shows that the proposed method is able to handle dis-occlusion via including global information. As shown in Fig. 11, LB matting[14] and KNN matting[12] fail in distinguishing background from the foreground when topology changes. Hair extracting is also a difficult task in vide matting. As shown in Fig. 12, the proposed method can handle the tiny hair of the object correctly in temporal and spatial domain. In contrast, the other four methods produce over smooth results when processing the tiny hair.

6 Conclusion

In this paper, we proposed a novel matting method based on patch division which saves memory consumption significantly, especially on high-resolution image and video. Instead of propagating alpha values to the whole image according to relationships among all pixels, by self-adaptive patch division, we apply the graph model to each patch with unknown pixels one by one. This framework is also extended to video matting with a new sampling method which reduces the time consumption. The proposed method significantly improves the memory efficiency while maintaining a high degree of visual accuracy. Based on the patch level framework, we would like to employ our method on GPU and parallel processing, to improve the time efficiency.

Acknowledgements We would like to thank the reviewers for their help in improving the paper. This work was partially supported by NSFC (61532003&61421003) and the Lenovo Outstanding Young Scientists Program.

References

1. Levin, A., Lischinski, D., Weiss, Y.: A closed form solution to natural image matting. In: CVPR, pp. 61–68 (2006)
2. Rhemann, C., Rother, C., Ravacha, A., Sharp, T.: High resolution matting via interactive trimap segmentation. In: CVPR, pp. 1–8 (2008)
3. http://www.alphamatting.com
4. http://www.videomatting.com
5. Shi, Y., Au, O., Pang, J., Tang, K., Sun, W., Zhang, H., Zhu, W., Jia, L.: Color clustering matting. In: ICME, pp. 1–6 (2013)
6. Shahrian, E., Rajan, D., Price, B., Cohen, S.: Improving image matting using comprehensive sampling sets. In: CVPR, pp. 636–643 (2013)
7. He, K., Rhemann, C., Rother, C., Tang, X., Sun, J.: A global sampling method for alpha matting. In: CVPR, pp. 2049–2056 (2011)
8. Karacan, L., Erdem, A., Erdem, E.: Image matting with KL-divergence based sparse sampling. In: ICCV, pp. 424–432 (2015)
9. Wang, J., Cohen, M.: Optimized color sampling for robust matting. In: CVPR, pp. 1–8 (2007)
10. Johnson, J., Rajan, D., Cholakkal, H.: Sparse codes as alpha matte. In: BMVC, vol. 1(3) (2014)
11. Shahrian, E., Rajan, D.,: Weighted color and texture sample selection for image matting. In: CVPR, pp. 718–725 (2012)
12. Chen, Q., Li, D., Tang, C.: Knn matting. In: CVPR, pp. 2175–88 (2013)
13. He, K., Sun, J., Tang, X.: Fast matting using large kernel matting laplacian matrices. In: CVPR, pp. 2165–2172 (2010)
14. Zheng, Y., Kambhamettu, C.: Learning based digital matting. In: ICCV, pp. 889–896 (2009)
15. Lee, P., Wu, Y.: Nonlocal matting. In: CVPR, pp. 2193–2200 (2011)
16. Aksoy, Y., Aydin, T.O., Pollefeys, M.: Designing effective inter-pixel information flow for natural image matting. In: CVPR (2017)
17. Tang, Z., Miao, Z., Wan, Y., Zhang, D.: Video matting via opacity propagation. Vis. Comput. **28**(1), 47–61 (2012)
18. Chen, X., Zou, D., Zhou, S.Z., Zhao, Q., Tan, P.: Image matting with local and nonlocal smooth priors. In: CVPR, pp. 1902–1907 (2013)
19. Wang, J., Cohen, M.: Image and video matting: a survey. Found. Trends Comput. Graph. Vis. **3**(2), 97–175 (2007)
20. Grady, L., Schiwietz, T., Aharon, S., Westermann, R.: Random walks for interactive alpha-matting. In: VIIP, pp. 423–429 (2005)
21. He, B., Wang, G., Shi, C., Yin, X., Liu, B., Lin, X.: Iterative transductive learning for alpha matting. In: ICIP, pp. 4282–4286 (2013)
22. Chuang, Y., Agarwala, A., Curless, B., Salesin, D., Szeliski, R.: Video matting. In: ACM SIGGRAPH, pp. 243–248(2002)
23. Li, D., Chen, Q., Tang, C.: Motion-aware knn laplacian for video matting. In: ICCV, pp. 3599–3606 (2013)
24. Sindeev, M., Anton, K., Carsten, R.: Alpha-flow for video matting. In: ACCV, pp. 438–452 (2012)
25. Johnson, J., Varnousfaderani, E.S., Cholakkal, H., Rajan, D.: Sparse coding for alpha matting. IEEE Trans. Image Process. **25**(7), 3032–3043 (2016)
26. Zou, D., Chen, X., Cao, G., Wang, X.: Video matting via sparse and low-rank representation. In: ICCV, pp. 1564–1572 (2015)
27. Choi, I., Lee, M., Tai, Y.: Video matting using multi-frame nonlocal matting laplacian. In: ECCV, pp. 540–553 (2012)
28. Cho, D., Tai, Y.W., Kweon, I.: Natural image matting using deep convolutional neural networks. In: ECCV. Springer, pp. 626–643 (2016)
29. Xu, N., Price, B., Cohen, S., Huang, T.: Deep image matting. In: CVPR (2017)
30. Cao, G., Li, J., He, Z., Chen, X.: Divide and conquer: a self-adaptive approach for high-resolution image matting. In: International Conference on Virtual Reality and Visualization (ICVRV), Sept, pp. 24–30 (2016)
31. Roweis, S., Saul, L.: Nonlinear dimensionality reduction by locally linear embedding. Science **290**(5500), 2323–2326 (2000)
32. Chen, X., Chen, D., Li, J., Cao, X.: Sparse dictionary learning for edit propagation of high-resolution images. In: CVPR, pp. 2854–2861 (2014)
33. Mairal, J., Bach, F., Ponce, J., Sapiro, G.: Online learning for matrix factorization and sparse coding. Mach. Learn. Res. **11**(1), 19–60 (2010)
34. Rhemann, C., Rother, J.W.C., Gelautz, M., Kohli, P., Rott, P.: A perceptually motivated online benchmark for image matting. In: CVPR, pp. 1826–1833 (2009)
35. Erofeev, M., Gitman, Y., Vatolin, D., Fedorov, A., Wang, J.: Perceptually motivated benchmark for video matting. In: BMVC, Sept, pp. 99.1–99.12 (2015)
36. Lee, S., Yoon, J., Lee, I.: Temporally coherent video matting. Graph. Models **72**(3), 25–33 (2010)

Guangying Cao is presently a Master candidate in the State Key Laboratory of Virtual Reality Technology and Systems, also in the School of Computer Science and Engineering, Beihang University. Her research interest is image and video processing.

Jianwei Li is presently a Ph.D. candidate in the State Key Laboratory of Virtual Reality Technology and Systems, also in the School of Computer Science and Engineering, Beihang University. His research interests are computer vision and image processing.

Xiaowu Chen received the Ph.D. degree at Beihang University in 2001. He is a professor in the State Key Laboratory of Virtual Reality Technology and Systems, also in the School of Computer Science and Engineering, Beihang University. His research interests include computer vision, computer graphics, virtual reality and augmented reality.

Zhiqiang He is currently the Senior Vice President of Lenovo Company and President of Lenovo Capital and Incubator Group. This group is responsible for exploring external innovation as well as accelerat-

ing internal innovation for Lenovo Group, leveraging Lenovo global resources, power of capital and entrepreneurship. Previously, Mr. He was the Chief Technology Officer and held various leadership positions in Lenovo, particularly in overseeing Lenovo's Research & Technology initiatives and systems. Mr. He is doctoral supervisor at Beihang University.